[0251] For example, a neural network can change the connections between nodes and the values assigned to the connections between nodes, but can never change the back-propagation rules that define how those values are determined. A genetic algorithm can create new genes and determine which genes are dominant, but cannot create a neural network. A logic program can combine existing logic to arrive at more powerful logical constructs, but cannot suddenly decide to represent these logical constructs as graphical representations (unless the programmer has already added this capability).

[0252] A program that describes itself (meta-implementation layer), describes its reason for existing in that manner (metamodel layer), and allows implementation to change dynamically (virtual implementation) lacks only the logic to do so. If a sufficiently powerful reasoning engine (neural network, genetic algorithm, or logic engine), were applied to such a language on a sufficiently powerful computer, such a program would be capable of rewriting itself to achieve the goals defined in the reasoning engine. A user could define these goals or these goals may be the result of operations conducted by the reasoning engine itself.

[0253] Virtual implementations of the present invention also allows for programming by contract functionality. For example, to enforce a contract asserting an operation cannot change any attributes of a parameter, a parameter implementation can create a read-only pass-through model and give that model to the operation. Even if the operation attempts to make a change to the parameter's attributes, it will fail because of the read-only protection. The operation can still perform all the normal read operations, since the read-only pass-through simply delegates read operations to the real parameter model. Not all programming languages have sufficient complexity to enforce contractual constraints like this one. Since a virtual implementation has access to its own description and behaves as an extensible language, it can add new functionality to enforce contract-programming ideas.

[0254] A virtual implementation of the present invention also allows for aspect programming. Due to the limited number of virtual implementation interfaces, it is possible to create a wrapper virtual implementation for each interface to add new aspects to all implementations. Adding new functionality to all aspects of a program is sometimes referred to as aspect programming. New aspects (like logging all attribute value changes) can be added systematically at runtime.

[0255] The meta-implementation layer of the present invention may blend parts of a virtual implementation with parts of other meta-implementations through the use of accessors. By selecting a virtual implementation for the model, then selecting accessors for database storage and accessors for compiled C++ source code, a meta-implementation can combine the advantages of each type of implementation to achieve the best balance of performance, flexibility, and maintenance.

[0256] As long as the appropriate accessors exist, the virtual model implementation can store its attributes in locations other than virtual attribute implementations and can perform operations in ways other than virtual operation implementations. This allows the virtual object to serve as a placeholder in the tool using the virtual implementation, while performing "non-virtual" data manipulation.

[0257] An example system might use meta-implementations to store all attributes data in a database. The constructor accessors would require enough information to create a record in a database and retrieve a primary key. The primary and foreign keys for this database record are stored in a virtual instance. All attribute changes to the virtual object are actually updates to the database, using the virtual instance to retrieve the primary key for the where clause. Most operation calls are stored procedure calls implemented as database-operation accessors. Some operation accessors may calls compiled C++ code written as part of a legacy system. These operation implementations perform calculations related to the data stored in the attributes. In order to make this system available over the Internet, a Java Servlet, Python module, or IIS plugin is used to access this meta-implementation layer and produce HTML.

[0258] Another approach to blending multiple different languages together is to create virtual implementations that are really references to a meta-implementation accessor. This allows the "true" meta-implementation layer to always point to a virtual implementation. Whenever functionality is needed in from a non-virtual implementation, an accessor implementation is used. This implementation simply delegates its responsibilities to a different meta-implementation provider.

[0259] For example, a virtual operation may contain several sub-operations. One of these sub-operations is an AccessorOperationImplementation that passes values to database-specific accessor to call a stored procedure. The OperationImplementation behaves exactly as the virtual operation expects, but performs its logic in the database. This is analogous to embedded SQL in a programming language.

[0260] In fact, more languages are being embedded into each other in today's development environment. Generally the embedded language is treated in such a way that the language into which it is embedded is unaware of it. Language integration allows all meta-implementations to interact with each other, fully aware of each other, but unconcerned about the specifics of implementation. Some of the most extreme cases of embedding languages into each other come from Active Server Pages (ASP) and Java Server Pages (JSP).

[0261] Both of these template-design languages contain a language that is executed on the server (Java for JSP, Visual Basic for ASP). This code can retrieve data from a database using embedded SQL. The results of that SQL query are translated to XML in order to be formatted by an XML Style sheet Language Template (XSLT). The XSLT produces an HTML document containing Cascading Style Sheets for style, HTML for content and markup, and JavaScript for dynamic content. The resulting ASP (or JSP) page contains all six languages in a single file: VB, SQL, XSLT, HTML, JavaScript, and CSS! While there are certainly better ways of performing this type of operation, it occurs frequently enough to warrant serious attention. Accessor implementations allow the mixing of an unlimited number of languages (and non-language software paradigms), as long as a meta-implementation accessor exists for each one. Each meta-implementation has its language completely separated while being integrated through the meta-implementation accessor interfaces and model descriptors.